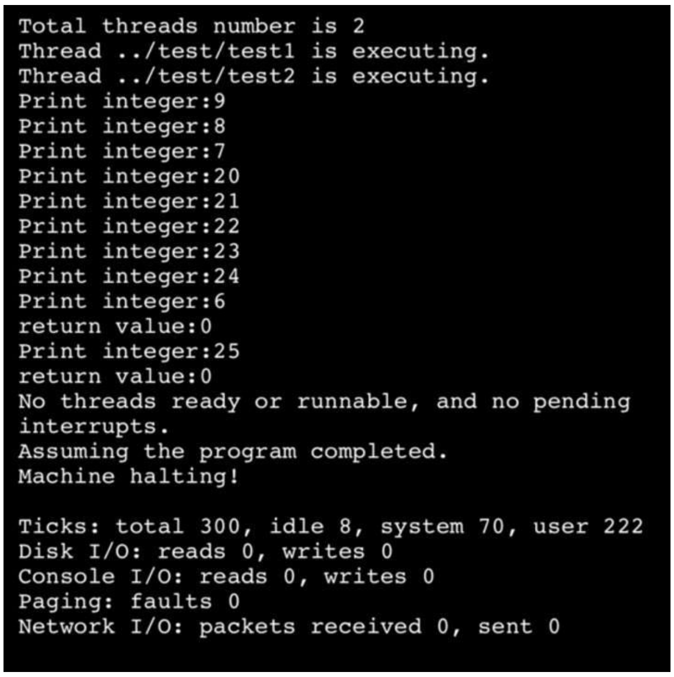
MP2 Demo

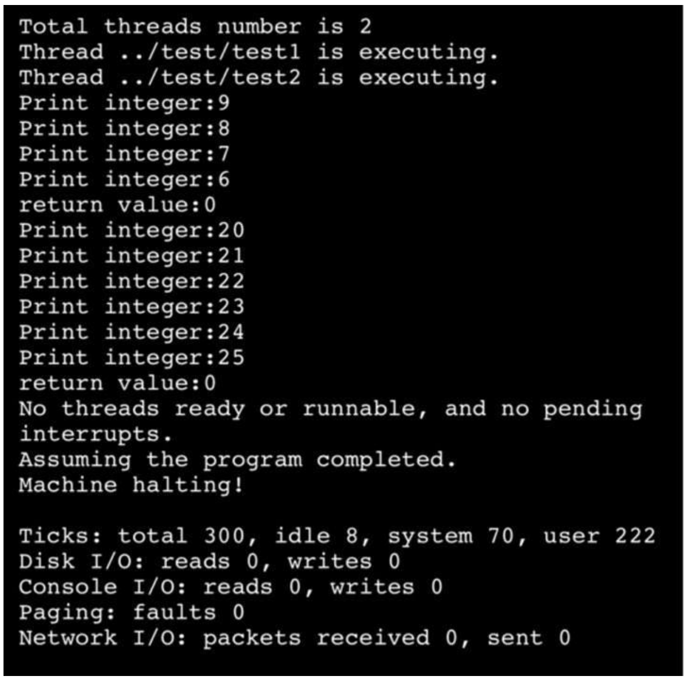
Without multi-programming



```
Total threads number is 2
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:6
return value:0
Print integer:25
return value:0
No threads ready or runnable, and no pending
interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 300, idle 8, system 70, user 222
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
```

After implement multi-programming



```
Total threads number is 2
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:25
return value:0
No threads ready or runnable, and no pending
interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 300, idle 8, system 70, user 222
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Code

```
classAddrSpace {
public:
AddrSpace();// Create an address space.
~AddrSpace();// De-allocate an address space
voidExecute(char *fileName);// Run the the program
// stored in the file "executable"
voidSaveState();// Save/restore address space-specific
voidRestoreState();// info on a context switch
boolusedPhyPage[NumPhysPages];  //用來記錄已經被使用過的 physical page
private:
TranslationEntry *pageTable;// Assume linear page table translation
// for now!
unsignedint numPages;// Number of pages in the virtual
// address space
boolLoad(char *fileName);// Load the program into memory
// return false if not found
voidInitRegisters();// Initialize user-level CPU registers,
// before jumping to user code
};

AddrSpace::AddrSpace()
{
```

```
/*  pageTable = new TranslationEntry[NumPhysPages];
  for (unsigned int i = 0; i < NumPhysPages; i++) {
  pageTable[i].virtualPage = i;  // for now, virt page # = phys page #
  pageTable[i].physicalPage = i;
//  pageTable[i].physicalPage = 0;
  pageTable[i].valid = TRUE;
//  pageTable[i].valid = FALSE;
  pageTable[i].use = FALSE;
  pageTable[i].dirty = FALSE;
  pageTable[i].readOnly = FALSE;
}
*/
  // zero out the entire address space
//  bzero(kernel->machine->mainMemory, MemorySize);
}

bool
AddrSpace::Load(char *fileName)
{
OpenFile *executable = kernel->fileSystem->Open(fileName);
NoffHeader noffH;
unsignedint size;

if(executable == NULL) {
cerr<< "Unable to open file " << fileName << "\n";
returnFALSE;
}
executable->ReadAt((char *)&noffH, sizeof(noffH), 0);
if((noffH.noffMagic != NOFFMAGIC) &&
(WordToHost(noffH.noffMagic) == NOFFMAGIC))
SwapHeader(&noffH);
ASSERT(noffH.noffMagic == NOFFMAGIC);

// how big is address space?
size= noffH.code.size + noffH.initData.size + noffH.uninitData.size
+ UserStackSize;// we need to increase the size
// to leave room for the stack
numPages= divRoundUp(size, PageSize);
//cout << "number of pagesof "<< fileName<< " is "<<numPages<<endl;

pageTable= new TranslationEntry[numPages];
for(unsigned int i = 0,j = 0; i < numPages; i++) {
```

```
pageTable[i].virtualPage = i;// for now, virt page # = phys page #
while(usedPhyPage[j++]==TRUE){}
usedPhyPage[j-1]=TRUE;
pageTable[i].physicalPage = j-1;
pageTable[i].valid = TRUE;
pageTable[i].use = FALSE;
pageTable[i].dirty = FALSE;
pageTable[i].readOnly = FALSE;
}
size= numPages * PageSize;
ASSERT(numPages <= NumPhysPages);// check we're not trying
// to run anything too big --
// at least until we have
// virtual memory

//DEBUG(dbgAddr, "Initializing address space: " << numPages << ", " << size);
// then, copy in the code and data segments into memory

if(noffH.code.size > 0) {
//DEBUG(dbgAddr, "Initializing code segment.");
//DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << noffH.code.size);
executable->ReadAt(
&(kernel->machine-
>mainMemory[pageTable[noffH.code.virtualAddr/PageSize].physicalPage*PageSize+(noffH.code.virtualAddr%PageSize)]),noffH.code.s
ize, noffH.code.inFileAddr);
}
if(noffH.initData.size > 0) {
//DEBUG(dbgAddr, "Initializing data segment.");
//DEBUG(dbgAddr, noffH.initData.virtualAddr << ", " << noffH.initData.size);
executable->ReadAt(
&(kernel->machine-
>mainMemory[pageTable[noffH.initData.virtualAddr/PageSize].physicalPage*PageSize+(noffH.code.virtualAddr%PageSize)]),noffH.init
Data.size, noffH.initData.inFileAddr);
}
deleteexecutable;// close file
returnTRUE;// success
}
```

組員:

劉家豪 00453136:50%

朱貴鴻 00453111:50%